

# COMPUTING SYMMETRY GROUPS OF POLYHEDRA

DAVID BREMNER, MATHIEU DUTOUR SIKIRIĆ, DMITRII V. PASECHNIK,  
THOMAS REHN, AND ACHILL SCHÜRMANN

ABSTRACT. Knowing the symmetries of a polyhedron  $P$  can be very useful for practical polyhedral computations as well for analysis of the structure of  $P$ .

We study the groups preserving the linear, projective and combinatorial structure of  $P$ . In each case we give algorithms to compute the symmetry group and discuss some practical experiences with these algorithms.

Our focus here is on  $\mathbb{R}^n$ ; we observe that some of the central notions do not admit a straightforward generalization to point configurations in complex projective spaces.

## CONTENTS

1. Introduction	1
2. Polyhedral Symmetry Groups	3
3. Computing $\text{Lin}_v(\mathcal{C})$	5
3.1. $\text{GL}_n(\mathbb{Z})$ symmetries	5
3.2. Symmetries of Integer Linear Programming Problems	7
3.3. Centralizer subgroups	7
3.4. Computing with vertex and edge colored graphs	9
4. Computing $\text{Proj}(\mathcal{C})$	9
5. Computing $\text{Comb}(\mathcal{C})$	13
References	14

## 1. INTRODUCTION

Symmetric polyhedra occur frequently in diverse contexts of mathematics. Polyhedra are central to the theory of Mathematical Optimization (Mathematical Programming), and main objects of study in linear and integer linear programming. Frequently studied symmetric polyhedra in applications such as transportation logistics or machine scheduling have names like “Travelling Salesman”, “Assignment” or “Matching”. For these and further examples we refer to [32] and the numerous references therein. Polyhedra play also prominent roles in other parts

---

The authors acknowledge the hospitality of Mathematisches Forschungsinstitut Oberwolfach (MFO) and Hausdorff Research Institute for Mathematics (HIM) in Bonn. Mathieu Dutour Sikirić was supported by the Humboldt Foundation. Dmitrii Pasechnik was supported by Singapore Ministry of Education ARF Tier 2 Grant MOE2011-T2-1-090.

of mathematics, e.g. in algebraic geometry, and in particular in the theory of toric varieties [6].

For the analysis of high dimensional polyhedra it is important to know their symmetries. Furthermore, for many important tasks in polyhedral computations, such as linear and integer linear programming, the representation conversion problem or volume computations, symmetry exploiting techniques are available (see [24] and [5]). Even commercial optimization software like [41] and [42] include some techniques for symmetry exploitation by now. To a large extent, the used methods depend on the kind of symmetry that is available and how it is presented. For instance, if we know the affine symmetry group of a polyhedron from a linear programming problem, we can reduce the problem dimension if the utility function is invariant under the group. In contrast to the full combinatorial symmetry group, the affine symmetry group has the advantage that it can practically be computed using only partial information, for instance, from a description of the polyhedron by linear inequalities.

In this paper we provide an overview on how to determine different symmetries of a polyhedron computationally. We provide a collection of computational recipes for the main polyhedral symmetry groups of interest. Our general philosophy is the translation into a problem of determining all the combinatorial automorphisms of a colored graph. Although these graph automorphism problems are not completely understood from a complexity theoretical point of view (see [16]), there exist sophisticated software tools for their practical solution [33, 36, 37]. All algorithms explained here are available within the GAP package `polyhedral` [34]. The linear group of a polyhedron can also be computed with the C++ tool `SymPol` [40]. Similar computational tasks for special classes of lattice polytopes are performed by `PALP` [38, 19]. `PALP` arose from the needs of the project to classify certain lattice polytopes called *reflexive polyhedra* [17, 18].

The paper is organized as follows. In Section 2 we define the three most important polyhedral symmetry groups and explain some of their relations among each other: The linear symmetry group, the projective symmetry group and the combinatorial symmetry group. In the following sections we consider each of them separately. We start with the linear symmetry group in Section 3. In Section 3.2 and Section 3.3 we deal with particular subgroups of the linear symmetry group that are important to Integer Linear Programming and applications in Computational Geometry of Positive Definite Quadratic Forms. In Section 4 we consider the projective symmetry group. We give a new characterization and from it derive a new algorithm for its computation. In our last Section 5, we consider a practical approach to the computation of the combinatorial symmetry group, which contains all the other groups.

## 2. POLYHEDRAL SYMMETRY GROUPS

A polyhedral cone  $\mathcal{C}$  in the vector space  $\mathbb{R}^n$  is defined as the set of vectors satisfying a finite number of linear (homogeneous) inequalities. By the Farkas-Minkowski-Weyl Theorem there exists a second (dual) description

$$\mathcal{C} = \{\lambda_1 v_1 + \dots + \lambda_p v_p : \lambda_i \in \mathbb{R}_{\geq 0}\}$$

with *generating vectors*  $v_1, \dots, v_p$  and *extreme rays*

$$R_i = \mathbb{R}_{\geq 0} v_i.$$

Whereas the extreme rays of  $\mathcal{C}$  are uniquely determined, the generating vectors are not.

A *face* of a polyhedral cone  $\mathcal{C}$  is the intersection of  $\mathcal{C}$  with a *supporting hyperplane*, that is, with a hyperplane  $\{x \in \mathbb{R}^n : a^t x = 0\}$  for some  $a \in \mathbb{R}^n$  such that  $\mathcal{C}$  is contained in the halfspace  $\{x \in \mathbb{R}^n : a^t x \geq 0\}$ . Faces are partially ordered by setwise inclusion, which gives a combinatorial structure that is called the *face lattice* of  $\mathcal{C}$ . The *dimension of a face* is defined as the dimension of the smallest linear subspace containing it. Without loss of generality, we assume that  $\mathcal{C}$  is full-dimensional, i.e. has dimension  $n$  and that it does not contain a non-trivial vector space. Facets of  $\mathcal{C}$  are faces of dimension  $n - 1$ . The set of facets and the set of extreme rays are uniquely determined by  $\mathcal{C}$  and the problem of passing from one description to the other is called the *dual description problem*.

A polytope  $P$  is the convex hull of a finite set of vectors  $(v_i)_{1 \leq i \leq M}$  in  $\mathbb{R}^n$ . By considering the vectors  $v'_i = (1, v_i)$  and the polyhedral cone  $\bar{\mathcal{C}}$  defined by  $(v'_i)_{1 \leq i \leq M}$ , we can actually embed  $P$  into  $\bar{\mathcal{C}}$  and translate the notions introduced for polyhedral cones to polytopes. For more information and background on polyhedra and polytopes we refer to [26].

By  $\mathcal{F}_k$  we denote the set of  $k$ -dimensional faces ( $k$ -faces) of  $\mathcal{C}$ . Such  $k$ -faces are identified with the set of extreme rays contained in them. The combinatorial symmetry group  $\text{Comb}(\mathcal{C})$  of  $\mathcal{C}$  is the group of all permutations of extreme rays that preserve  $\mathcal{F}_k$  for all  $0 \leq k \leq n - 1$ . So in particular,  $\text{Comb}(\mathcal{C})$  is a subgroup of the symmetric group  $\text{Sym}(p)$  on  $p$  elements.

It is well known that  $\text{Comb}(\mathcal{C})$  is actually determined by  $\mathcal{F}_{n-1}$ :  $\text{Comb}(\mathcal{C})$  is isomorphic to the automorphism group of the bipartite facet-ray-incidence graph. Indeed, there is generally no simpler way to compute  $\text{Comb}(\mathcal{C})$ , as this problem is graph isomorphism complete even for simple or simplicial polytopes (see [14]). For the construction of this bipartite graph we must know both, the extreme rays and the facets of  $\mathcal{C}$ . However, in practice, usually only one of these descriptions is known.

Let  $\text{GL}(\mathcal{C})$  be the group of invertible matrices  $A \in \text{GL}_n(\mathbb{R})$ , which induce a projective symmetry of  $\mathcal{C}$ , that is  $A\mathcal{C} = \mathcal{C}$ . The action of  $\text{GL}(\mathcal{C})$  on  $\mathcal{C}$  induces a permutation of the extreme rays of  $\mathcal{C}$ . We call the resulting permutation group the *projective symmetry group*  $\text{Proj}(\mathcal{C})$  of  $\mathcal{C}$ , which is a subgroup of  $\text{Comb}(\mathcal{C})$ . In other words, elements of  $\text{Proj}(\mathcal{C})$  are permutations  $\sigma \in \text{Sym}(p)$  for which there exist matrices  $A \in \text{GL}(\mathcal{C})$  such that  $AR_i = R_{\sigma(i)}$  for  $1 \leq i \leq p$ .

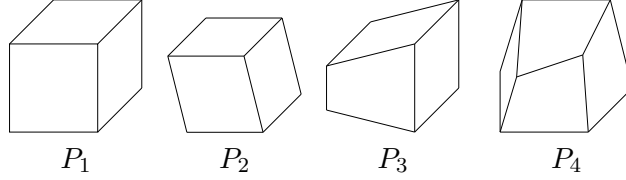


FIGURE 1. 4 polytopes in Euclidean space. The classes of equivalence under linear, projective and combinatorial equivalence are  $(\{P_1, P_2\}, \{P_3\}, \{P_4\})$ ,  $(\{P_1, P_2, P_3\}, \{P_4\})$  and  $(\{P_1, P_2, P_3, P_4\})$  respectively.

Note, however, that  $\text{GL}(\mathcal{C})$  and  $\text{Proj}(\mathcal{C})$  are not isomorphic since the kernel  $K$  of the homomorphism  $\text{GL}(\mathcal{C}) \rightarrow \text{Proj}(\mathcal{C})$  is non-trivial. It contains for instance all dilations. In group-theoretic terms,  $\text{GL}(\mathcal{C}) \cong K \times \text{Proj}(\mathcal{C})$ , cf. Theorem 4. In a more general setting of symmetries of a configuration of  $p$  points  $R_i$  in a projective space over a field (e.g. over  $\mathbb{C}$ ), the analog of the group  $\text{Proj}(\mathcal{C})$  does not need to exist, as  $K$  does not need to be split from  $\text{GL}(\mathcal{C})$ . See Remark 5 for a simple example of the latter, and [30, 13] for the related group-theoretic notions.

Fixing a set  $v = \{v_i\}_{1 \leq i \leq p}$  of generators for the extreme rays  $\{R_i\}_{1 \leq i \leq p}$ , we obtain in a similar way the *linear symmetry group*  $\text{Lin}_v(\mathcal{C})$  of  $\mathcal{C}$  (with respect to  $v$ ): It consists of all permutations  $\sigma \in \text{Sym}(p)$  for which there exist matrices  $A \in \text{GL}(\mathcal{C})$  such that  $Av_i = v_{\sigma(i)}$  for  $1 \leq i \leq p$ . Clearly,  $\text{Lin}_v(\mathcal{C})$  is always isomorphic to a subgroup of  $\text{GL}(\mathcal{C})$ , which we call  $\text{GL}_v(\mathcal{C})$ . In the particular case of a polytope  $P$  with an associated cone  $\mathcal{C}$  generated by  $(v'_i = (1, v_i))_{1 \leq i \leq p}$  the group  $\text{GL}_v(\mathcal{C})$  is actually the group of affine transformations preserving  $P$ .

For every polyhedral cone  $\mathcal{C}$  and every set of generators  $v$  we have

$$\text{Lin}_v(\mathcal{C}) \leq \text{Proj}(\mathcal{C}) \leq \text{Comb}(\mathcal{C}).$$

Both inclusions can be strict and we can define the corresponding notions of equivalence. In Figure 1 we give examples of those notions for the cube.

In Section 4 we prove that the projective symmetry group  $\text{Proj}(\mathcal{C})$  can be realized as  $\text{Lin}_v(\mathcal{C})$  for a suitable choice of vectors  $v$ . However, in [4, 12, 26] some polytopes whose combinatorial symmetries cannot be realized as projective symmetries are given.

Using the implementation in [34] we have compared the linear, projective and combinatorial symmetry group of 4313 polytopes available from the web page of A. Paffenholz [43]. For these examples, only in one case is the projective symmetry group larger than the linear symmetry group. This example was the one obtained by applying the construction  $E_2$  [26] to the 4-simplex; it is projectively equivalent to the dual of the Johnson polytope  $J(5, 2)$ . For 75 of the 4313 examples, the combinatorial symmetry group is larger than the projective symmetry group. The additional symmetries are in most cases a factor of 2 but reached in two cases a factor of 36.

### 3. COMPUTING $\text{Lin}_v(\mathcal{C})$

In this section we give algorithms to compute the linear symmetry group  $\text{Lin}_v(\mathcal{C})$  and certain subgroups occurring in applications like combinatorial optimization.

The linear symmetry group is easier to compute in practice than either the projective or combinatorial group, at least in the typical situation where we have only a generator representation (or only an inequality representation) for the input. Furthermore, the computation of the linear group is used as a subroutine in our algorithms to compute the projective and the combinatorial symmetry groups.

One practical method for computing  $\text{Lin}_v(\mathcal{C})$  is to define a positive definite matrix

$$(1) \quad Q = \sum_{i=1}^p v_i v_i^T$$

and the graph  $G(v)$  on  $p$  vertices  $\{1, \dots, p\}$  with vertex and edge colors  $w_{i,j} = v_i^T Q^{-1} v_j$ . As shown in [5] the linear group  $\text{Lin}_v(\mathcal{C})$  is then the group of permutations that preserve the colors. In the particular case of a polytope  $P \subset \mathbb{R}^n$  generated by  $(v_i)_{1 \leq i \leq p}$  with associated polyhedral cone  $\mathcal{C}$  generated by  $(v'_i = (1, v_i))_{1 \leq i \leq p}$  the matrix  $Q^{-1}$  allows to define a Euclidean scalar product on  $\mathbb{R}^n$  for which  $\text{GL}_v(\mathcal{C})$  is the group of affine isometries.

By using the methods presented in Section 3.4 one can compute  $\text{Lin}_v(\mathcal{C})$  for polytopes with a few thousand vertices. For polytopes with a large vertex set some reduction may be necessary. One idea used in [22], for which the polytope has about  $10^8$  vertices, is to compute the stabilizer of a vertex.

In high dimensions a key bottleneck is the computation of the inverse of the matrix  $Q$ . One approach to the problem is to compute the inverse using double precision. A tolerance number  $tol$  has to be chosen and values of  $(Q^{-1})_{ij}$  which are within  $tol$  have to be grouped. One then computes the automorphism group of the colored graph for the grouped colors and checks if the graph automorphisms can be represented by matrices. If they cannot, then  $tol$  has to be decreased or double precision is not enough.

**3.1.  $\text{GL}_n(\mathbb{Z})$  symmetries.** The approach described in the previous section to compute  $\text{Lin}_v(\mathcal{C})$  as a graph automorphism works reasonably well in most cases. However, for polyhedra with many generators this approach may not be applicable because the complete graph considered is too large.

For certain polyhedra, the linear symmetries are realized as  $\text{GL}_n(\mathbb{Z})$  matrices (integer matrices of determinant  $\pm 1$ ). A particular class of polytopes that we encountered with this property are so called *consecutive ones polytopes* (see, for example, [25]). These arise as the convex hull of  $m \times n$  matrices with 0, 1 entries satisfying a consecutive ones property. Because these polytopes have about  $2^{m \cdot n}$  vertices in dimension  $mn$ , the graph construction for computing symmetries is infeasible even for small  $m$  and  $n$ . The linear symmetries can nevertheless be obtained very quickly in small dimensions using the approach discussed in this section.

We now describe a method to compute the group  $\mathrm{GL}_v(\mathcal{C}, \mathbb{Z}) := \mathrm{GL}_v(\mathcal{C}) \cap \mathrm{GL}_n(\mathbb{Z})$ . As discussed above, in some cases the induced permutations generate  $\mathrm{Lin}_v(\mathcal{C})$ . Let the representatives  $(v_i)_{1 \leq i \leq p}$  be integral. We conclude that  $G_Q = \{A \in \mathrm{GL}_n(\mathbb{Z}) : A^T Q A = Q\}$  (with  $Q$  as in (1)) contains  $\mathrm{GL}_v(\mathcal{C}, \mathbb{Z})$ . In particular we obtain  $\mathrm{GL}_v(\mathcal{C}, \mathbb{Z})$  as the setwise stabilizer of  $\{v_1, \dots, v_p\}$  in  $G_Q$ . The group  $G_Q$  is the automorphism group of a lattice, that is of a discrete additive subgroup of the integral vectors. The matrix  $Q$  is an integral gram matrix of a lattice basis. The group  $G_Q$  can thus be computed with the algorithm of Plesken and Souvignier [28]. It remains to perform a stabilizer computation in this matrix group. In principle, the Plesken-Souvignier algorithm can be adapted to include this stabilization. As the matrices  $A$  are generated row by row in a backtrack algorithm, we can check whether the current set of rows of  $A$  stabilizes a projection of  $\{v_1, \dots, v_p\}$  accordingly. If it violates this stabilization property, we may discard the entire candidate branch.

The Plesken-Souvignier algorithm computes a set  $S \subset \mathbb{Z}^n$  of short lattice vectors, which may be a difficult task for itself. Then a backtrack search on  $S$  is employed to compute  $G_Q$ . If we briefly ignore the computational cost of  $S$ , the lattice approach has the advantage to work with an  $n \times n$  instead of a  $p \times p$  matrix. Thus for polyhedra which are generated by many rays and for which  $S$  is not too difficult to compute this may be a viable alternative.

Theoretically, this method could be used to compute the linear symmetries of any cone  $\mathcal{C}$ . In particular, if  $\{v_1, \dots, v_p\} \subset \mathbb{Z}^n$  and contains the standard basis  $e_1, \dots, e_n$ , the groups  $\mathrm{GL}_v(\mathcal{C}, \mathbb{Z})$  and  $\mathrm{GL}_v(\mathcal{C})$  are the same. If this condition is not satisfied by the input, we may transform  $\mathcal{C}$  linearly without altering its linear symmetries.

In some applications the goal is to find some  $\mathrm{GL}_n(\mathbb{Z})$  subgroup of  $\mathrm{GL}_v(\mathcal{C})$ , rather than the full linear symmetry group. One such application occurs in [10] when computing the Delaunay tessellations of a  $n$ -dimensional lattice  $L$ . We denote by  $\mathrm{Isom}(P)$  the group of isometries of a Delaunay polytope  $P$  and by  $\mathrm{Isom}_L(P)$  the group of isometries of  $P$  that also preserve the lattice  $L$ . One simple technique to find  $\mathrm{Isom}_L(P)$  is to iterate over  $\mathrm{Isom}(P)$  and keep the elements belonging to  $\mathrm{Isom}_L(P)$ . The finite group  $\mathrm{Aut}(L)$  of isometries of  $L$  preserving  $0$  is identified with the group of isometries of the quotient  $\mathbb{R}^n/L$ . The center  $c$  of the empty sphere of  $P$  is expressed as  $\frac{v}{m}$  with  $v \in L$  and the group  $\mathrm{Isom}_L(P)$  is identified with the stabilizer of  $c \in \mathbb{R}^n/L$  by  $\mathrm{Aut}(L)$ . For any divisor  $d$  of  $m$  we can consider the stabilizer in  $\mathbb{R}^n/L'$  with  $L' = L/d$  and the factorization of  $m$  gives a sequence of stabilizers that converges to  $\mathrm{Isom}_L(P)$ . However, the most powerful technique to compute  $\mathrm{Isom}_L(P)$  in practice is to apply the Plesken-Souvignier algorithm to the following homogenization: We define a  $(n+1)$ -dimensional lattice  $L'$  spanned by all  $(0, v)$  and  $(0, v - c)$  for  $v \in L$ . The automorphism group of  $L'$  contains an index 2 subgroup isomorphic to  $\mathrm{Isom}_L(P)$ , see [10] for more details and [34] for an implementation of this technique which is very similar to the previous use of the Plesken-Souvignier algorithm.

In the general case of a polytope  $P$  for which we want the whole group  $\text{GL}_v(P, \mathbb{Z})$  the above techniques do not apply. Apart from the method of iterating over all elements, one technique could be to add elements to the vertex set of  $P$  until we get a set that spans  $\mathbb{Z}^n$ .

**3.2. Symmetries of Integer Linear Programming Problems.** In integer linear programming, one optimizes over the intersection of a polyhedron with the integer lattice. From a mathematical point of view, the natural symmetries thus preserve the polyhedron and the integer lattice, i.e. are  $\text{GL}_n(\mathbb{Z})$  symmetries. As far as we know, these general symmetries are not used in existing integer optimization software. Instead, the common practice is to consider only *coordinate symmetries*, i.e. permutations of coordinates that are automorphisms of the polyhedron. These symmetries turn out to be easier to compute, and also straightforward to work with in integer programming solvers.

Several authors have been concerned with ways to compute coordinate symmetries, by reducing the problem to a graph automorphism problem (see [29, 31, 3, 23]). Coordinate symmetries are isomorphic to the automorphisms of the following bipartite graph. Its vertex set is the union  $\{v_1, \dots, v_p\} \cup \{x_1, \dots, x_n\}$  of generators (coming from inequalities) and variables. Between each pair  $v_i$  and  $x_j$  we add an edge colored by the coefficient of variable  $x_j$  in generator  $v_i$ . The input is transformed to a bipartite edge colored graph, which is simpler than the complete colored graph required for more general symmetries. In integer programming we also have an objective function, which has to be considered for symmetry computation. We can deal with this by coloring the graph vertices that correspond to the variables  $x_1, \dots, x_n$  by their respective objective coefficient.

Using this graph and transformation techniques detailed in Section 3.4, the permutation symmetries of 353 polytopes from mixed integer optimization were computed in [27]. Some of the larger instances, which had more than one million variables or facets, were still computationally tractable. In 208 polytopes a non-trivial symmetry group was found. For the 50 smallest problems, with dimension less than 1500, we computed also the linear symmetry group. We found that in 6 out of these 50 cases the linear symmetry group is larger than the permutation symmetry group. All these linear symmetries are realized by integral matrices.

**3.3. Centralizer subgroups.** We now give an algorithm for centralizer subgroups that is useful particularly in applications in the Geometry of Numbers; several examples follow the proof. For a given set  $\mathcal{B} \subset M_n(\mathbb{R})$  we want to find the group

$$\text{GL}_v(\mathcal{C}, \mathcal{B}) = \{A \in \text{GL}_v(\mathcal{C}) \text{ s.t. } AB = BA \text{ for } B \in \mathcal{B}\}.$$

Without loss of generality we may assume that the set  $\mathcal{B}$  is linearly independent, contains the identity matrix and is written as  $\{B_1, \dots, B_r\}$  with  $B_1 = I_n$ . We denote by  $\text{Lin}_v(\mathcal{C}, \mathcal{B})$  the corresponding isomorphic permutation group which is a subgroup of  $\text{Lin}_v(\mathcal{C})$ .

**Theorem 1.** If  $\mathcal{B} = \{B_1, \dots, B_r\}$  is a set of  $n \times n$ -matrices with  $B_1 = I_n$  then the group  $\text{Lin}_v(\mathcal{C}, \mathcal{B})$  is the group of permutations  $\sigma$  preserving the directed colored



graph with edge and vertex colors

$$w_{ij} = (v_i^T Q^{-1} B_1 v_j, \dots, v_i^T Q^{-1} B_r v_j)$$

with

$$Q = \sum_{i=1}^p v_i v_i^T$$

*Proof.* If  $A \in \text{GL}_v(\mathcal{C}, \mathcal{B})$  then  $AB_i = B_i A$  and  $Av_i = v_{\sigma(i)}$ . Hence one gets by summation that  $AQA^T = Q$  or equivalently  $Q^{-1} = A^T Q^{-1} A$  (obtained from  $Q^{-1} = (A^T)^{-1} Q^{-1} A^{-1}$  by left and right multiplication). So, if one writes  $h_{ij}^k = v_i^T Q^{-1} B_k v_j$  then one gets

$$\begin{aligned} h_{ij}^k &= v_i^T Q^{-1} B_k v_j \\ &= v_i^T A^T Q^{-1} A B_k v_j \\ &= (Av_i)^T Q^{-1} B_k (Av_j) \\ &= h_{\sigma(i)\sigma(j)}^k \end{aligned}$$

So, any  $A$  induces a permutation of the  $p$  vertices preserving the vector edge color  $w_{ij}$ .

Suppose now that  $\sigma \in \text{Sym}(p)$  satisfies  $w_{ij} = w_{\sigma(i)\sigma(j)}$ . Then, since  $B_1 = I_n$  we have  $v_i^T Q^{-1} v_j = v_{\sigma(i)}^T Q^{-1} v_{\sigma(j)}$ . By an easy linear algebra computation (see [9, 5] for details) we get that there exists  $A \in \text{GL}_n(\mathbb{R})$  such that  $Av_i = v_{\sigma(i)}$ . If one writes  $w_i = Q^{-1} Av_i$  then

$$\begin{aligned} w_i^T AB_k v_j &= v_i^T A^T Q^{-1} AB_k v_j \\ &= v_i^T Q^{-1} B_k v_j \\ &= h_{ij}^k \\ &= h_{\sigma(i)\sigma(j)}^k \\ &= v_{\sigma(i)}^T Q^{-1} B_k v_{\sigma(j)} \\ &= v_i^T A^T Q^{-1} B_k Av_j \\ &= w_i^T B_k Av_j \end{aligned}$$

Since the families  $(v_j)$  and  $(w_i)$  span  $\mathbb{R}^n$  we get  $AB_k = B_k A$ .  $\square$

There are many contexts where the above theorem is useful. For example if one wishes to find the group of elements  $A \in \text{GL}_n(\mathbb{Q}[\sqrt{5}])$  then one way is to express the elements as elements of  $\text{GL}_{2n}(\mathbb{Q})$  that commute with the multiplication by  $\sqrt{5}$ . This is very useful when working with Humbert forms [1] whose symmetry group in  $\text{GL}_n(\mathbb{Z}[\sqrt{5}])$  correspond to a small subgroup of the full group in  $\text{GL}_{2n}(\mathbb{Z})$ .

Another such example is if one wishes to find the elements belonging to  $\text{GL}_n(\mathbb{H})$  with  $\mathbb{H}$  the Hamilton's quaternions.  $\text{GL}_n(\mathbb{H})$  acts on  $\mathbb{H}^n$  by multiplication on the left and it is characterized in  $\text{GL}_{4n}(\mathbb{R})$  by the fact that it commutes with the Hamiltonian multiplication on the right.

Another example is if one wishes to compute the group  $\text{GL}_v(\mathcal{C}, W)$  of elements of  $\text{GL}_n(\mathbb{R})$  preserving a polytope  $\mathcal{C}$  and a vector space  $W$ . Any such element will be an isometry for the scalar product defined by  $Q^{-1}$  in the proof of Theorem 1 and so will commute with the orthogonal projection on  $W$ .



One would wish for a similar characterization for elements of  $\text{GL}_v(\mathcal{C})$  that preserve a set  $\mathcal{B}$  setwise by conjugation and so compute normalizer groups. But we do not think that a similar characterization is possible.

**3.4. Computing with vertex and edge colored graphs.** Many of the algorithms presented above depend on the computation of the automorphism group of a graph whose vertices and/or edges are colored. The complexity of the graph isomorphism problem is uncertain as it is one of the rare problems in NP which is neither known to be NP-complete nor in P.

For practical computation there exists graph isomorphism software (see [37, 36, 33, 39]) that can compute automorphism groups of graphs. Such programs usually use the partition backtrack algorithm and can compute the automorphism groups of large graphs but their run time is exponential in the worst case. These programs usually cannot handle edge colored graphs and suffer from a performance penalty when using digraphs. Hence, one needs reduction techniques.

There are several techniques for reducing an edge colored graph to a vertex colored graph (i.e. a complete graph with only two edge colors). One transformation described in [31] replaces each  $c$ -colored edge  $\{a, b\}$  with an intermediate  $c$ -colored vertex  $m$ , which has edges connecting it to  $a$  and  $b$ . The obtained graph has  $n + n(n-1)/2$  vertices which makes the transformation expensive. For the bipartite edge colored graphs that occur in Section 3.2 this can be improved by an idea given in [29]. Instead of adding intermediate vertices for all edges, we combine some of those with the same color. Define the bipartition as  $(S, S')$ . For each  $i \in S$  let  $X_{i,c} \subseteq S'$  be the set of vertices which are incident to  $i$  with an edge of color  $c$ . Then it is enough to introduce an intermediate  $c$ -colored vertex  $m$  with edges to  $i$  and to all elements of  $X_{i,c}$ . For many integer optimization problems these sets  $X_{i,c}$  are often large, thus the number of vertices in the graph is usually substantially reduced. If  $|S| > |S'|$ , it may be advantageous to combine edges the other way around.

For general edge-colored graphs we use the following method proposed in the user manual of `nauty`, [37]. Suppose that we have  $M$  colors. Then any color can be expressed as a 0/1 word of length  $\lceil \log_2(M) \rceil$ . Hence, the automorphism group can be obtained from the superposition of  $\lceil \log_2(M) \rceil$  vertex colored graphs and so from a graph with  $p \lceil \log_2(M) \rceil$  vertices. This solution has good complexity estimates but the preceding method is often the best for the bipartite graphs from integer optimization (see [27]).

Colored digraphs can be transformed into colored graphs with twice the number of vertices. A vertex  $a$  corresponds to a pair  $\{a, a'\}$  and a directed edge  $(a, b)$  to an undirected edge  $(a, b')$ . Let  $c$  and  $c'$  two colors that do not occur as directed edge color. We assign edges  $(a, b)$ , respectively  $(a', b')$  the color  $c$ , respectively  $c'$ .

#### 4. COMPUTING $\text{Proj}(\mathcal{C})$

For a given polyhedral cone  $\mathcal{C}$ , the group  $\text{Proj}(\mathcal{C})$  is the group of permutations of extreme rays that are induced by a linear transformation of  $\mathcal{C}$ . We give a method

allowing the computation of the projective group in practice. It is based on a decomposition for polyhedral cones and on some linear algebra tests.

A polyhedral cone  $\mathcal{C}$  generated by rays  $E = (R_i)_{1 \leq i \leq p}$  in a vector space  $V$  is said to be *decomposable* if there exist two non-empty subspaces  $V_1, V_2$  such that  $E = (E \cap V_1) \cup (E \cap V_2)$  and  $V = V_1 \oplus V_2$ .

**Theorem 2.** For a polyhedral cone  $\mathcal{C}$  generated by  $E = (R_i)_{1 \leq i \leq p}$  in a vector space  $V$ , there is a unique decomposition

$$V = \oplus_{1 \leq k \leq h} V_k$$

with

$$E = \cup_{1 \leq k \leq h} E \cap V_k$$

and the cone  $\mathcal{C}_k$  generated by  $E \cap V_k$  being non-decomposable. This decomposition can be computed in time  $O(pn^2)$ .

*Proof.* The existence of the decomposition is obvious since the vector space  $V$  is finite dimensional and so the decomposition process has to end at some point. The uniqueness follows immediately from the fact that if there are two decompositions by  $V_k$  and  $V'_l$  then the family of subspaces  $V_k \cap V'_l$  also defines a decomposition.

The method for computing a decomposition is the following. First select some generators  $v_i$  of  $R_i$ . Then find (e.g. by Gaussian elimination) a  $n$ -element set  $S \subset \{1, \dots, p\}$  such that  $(v_i)_{i \in S}$  is a basis of  $V$ . Every vector  $v_i$  for  $1 \leq i \leq p$  is then written as  $v_i = \sum_{k \in S} \alpha_{k,i} v_k$ . The supports

$$S_i = \{k \in S : \alpha_{k,i} \neq 0\}$$

determine the edges of an hypergraph on  $n$  vertices. The connected components of this hypergraph correspond to the summands of the decomposition of  $V$ . Both computing  $S$  and finding the connected components can be done in  $O(pn)$  time.  $\square$

The above decomposition allows one to determine the projective symmetry group. We say that two cones have the same *projective isomorphism type* if there is a bijective linear map between them.

**Theorem 3.** For a polyhedral cone  $\mathcal{C}$  generated by rays  $E = (R_i)_{1 \leq i \leq p}$  in a vector space  $V$ , decomposed into  $V = \oplus_{1 \leq k \leq h} V_k$ , define  $\mathcal{C}_k$  to be the cone generated by  $E \cap V_k$ .

If the projective isomorphism type  $j$  for  $1 \leq j \leq q$  occurs  $n_j$  times among the  $\mathcal{C}_k$  then we have the equality

$$\text{Proj}(\mathcal{C}) = \prod_{j=1}^q \text{Wr}(\text{Sym}(n_j), \text{Proj}(\mathcal{C}_{k_j}))$$

where  $\text{Wr}$  stands for wreath product and  $\mathcal{C}_{k_j}$  a representative for the  $j$ th type.

*Proof.* Suppose that we have an element  $f \in \text{Proj}(\mathcal{C})$ . This element permutes the  $\mathcal{C}_k$  but must also preserve the projective isomorphism types. Hence, it belongs to the above mentioned product. The reverse inclusion is trivial.  $\square$

We now expose a method for computing the projective symmetry group of a non-decomposable polyhedral cone  $\mathcal{C}$ . Combined with the above theorem, this

will allow us to compute the projective symmetry group of arbitrary polyhedra. We first present the following structural result:

**Theorem 4.** Suppose  $\mathcal{C}$  is a non-decomposable polyhedral cone generated by rays  $(R_i)_{1 \leq i \leq p}$ . Then

(i) We have the isomorphism

$$\mathrm{GL}(\mathcal{C}) \cong \mathbb{R}_+^* \times \mathrm{Proj}(\mathcal{C}).$$

(ii) There exist vectors  $v_i$  such that  $R_i = \mathbb{R}_+ v_i$  and

$$\mathrm{Lin}_v(\mathcal{C}) = \mathrm{Proj}(\mathcal{C}).$$

*Proof.* To show (i), we have to determine the kernel  $K$  of the homomorphism  $\mathrm{GL}(\mathcal{C}) \rightarrow \mathrm{Proj}(\mathcal{C})$  and show that  $K$  splits from  $\mathrm{GL}(\mathcal{C})$ , i.e.  $\mathrm{GL}(\mathcal{C}) \cong K \times \mathrm{Proj}(\mathcal{C})$ . For  $A \in K$  we have  $AR_i = \alpha_i R_i$  with  $\alpha_i > 0$ . If the values  $\alpha_i$  were not all the same then this would automatically give a decomposition of the space (since each class of rays with the same multiplier will be subdimensional) contradicting the non-decomposability of  $\mathcal{C}$ . Thus  $K = \{\lambda I \mid \lambda \in \mathbb{R}_+^*\}$ . Therefore  $\mathrm{GL}(\mathcal{C})$  is a central extension  $K \cdot \mathrm{Proj}(\mathcal{C})$  of  $\mathrm{Proj}(\mathcal{C})$  by  $K$ , i.e.  $K$  is normal in  $\mathrm{GL}(\mathcal{C})$  and in particular lies in the center of  $\mathrm{GL}(\mathcal{C})$  (see e.g. [13, (11.8)]). A classical theorem due to I. Schur states that every non-split central extension (i.e. a central extension  $A.B$  which is not isomorphic to  $A \times B$ ) of a finite group  $G$  can be obtained as a homomorphic image of  $M(G).G$ , where  $M(G)$  is a finite Abelian group, called the *Schur multiplier* (cf. [30] or [13, (11.17)]) of  $G$ . Thus  $K \geq M'$ , where  $M'$  is a quotient group of the Schur multiplier  $M(\mathrm{Proj}(\mathcal{C}))$  of  $\mathrm{Proj}(\mathcal{C})$ . As  $K$  is torsion-free (i.e. has no non-identity elements of finite order), this implies that  $M' = 1$ , and  $\mathrm{GL}(\mathcal{C}) = K \times \mathrm{Proj}(\mathcal{C})$ , as claimed.

To show (ii) we can, by (i), identify  $\mathrm{Proj}(\mathcal{C})$  with a subgroup  $H$  of  $\mathrm{GL}(\mathcal{C})$ . This identification is unique, as  $\mathrm{Proj}(\mathcal{C})$  is the subgroup of elements of finite order. For each  $\mathrm{Proj}(\mathcal{C})$ -orbit  $O_k$  of extreme rays we choose a representative ray  $R_k \in O_k$  and a vector  $v_k$  such that  $R_k = \mathbb{R}_+ v_k$ . For each element  $R \in O_k$  we can find an  $f \in H$  such that  $R = f(R_k)$ . If there is another  $f' \in H$  such that  $R = f'(R_k)$  then  $h(v_k) = C v_k$  with  $h = f^{-1} f'$  and  $C > 0$ . Since  $h \in H$ , it has finite order, and thus  $C = 1$ . This means that  $f(v_k)$  is uniquely defined. It is then clear that for this choice of generators  $\mathrm{Proj}(\mathcal{C}) = \mathrm{Lin}_v(\mathcal{C})$ .  $\square$

**Remark 5.** The conclusions of Theorem 4 cease to hold in a more general setting of a configuration of points in a projective space over  $\mathbb{C}$ . In the following we construct a counterexample.

First we take the group  $G$  characterized as  $2 \cdot S_4^-$  (namely, number 28 in GAP database of small groups [35]) and its faithful 2-dimensional representation over  $\mathbb{C}$ . The center of  $G$  in this representation is  $\pm \mathrm{Id}_2$ . We then take an element of order 8 in  $G$  and compute one of its eigenspaces, corresponding to an 8-th primitive root of unity. There are 6 images of the eigenspace under  $G$ . Thus we obtain a transitive action of  $G$  on a 6-tuple of lines and so on 6 points in  $P^1(\mathbb{C})$ . The action on the 6 lines defines a group  $S_4$ . But  $G$  is not isomorphic to  $2 \times S_4$ .

**Theorem 6.** Suppose  $\mathcal{C}$  is a non-decomposable cone generated by rays  $(R_i)_{1 \leq i \leq p}$  in  $\mathbb{R}^n$ . Testing if a permutation  $\sigma \in \text{Sym}(p)$  belongs to  $\text{Proj}(\mathcal{C})$  can be done by solving a linear system with  $np$  equations and  $p$  unknowns.

*Proof.* Let us slightly abuse notation and denote by  $V_\mu = (v_{\mu(1)}, \dots, v_{\mu(p)})$  the matrix with columns being a set of generators for the  $R_i$ , i.e.  $R_i = \mathbb{R}_+ v_i$ , for  $1 \leq i \leq p$ , permuted by a permutation  $\mu$ . Respectively, let  $U_\mu$  denote the submatrix of  $V_\mu$  consisting of its first  $n$  columns, and  $V := V_{\text{id}}$ ,  $U := U_{\text{id}}$ . Without loss of generality,  $U$  is invertible.

The sought matrix  $A \in \text{GL}(\mathcal{C})$ —a preimage of  $\sigma$ —must satisfy  $Av_i = \alpha_i v_{\sigma(i)}$ ,  $\alpha_i > 0$ , for each  $1 \leq i \leq p$ . In the matrix form this can be written as  $AV = V_\sigma \text{diag}(\alpha_1, \dots, \alpha_p)$ . In particular,  $AU = U_\sigma \text{diag}(\alpha_1, \dots, \alpha_n)$ , implying

$$(2) \quad A = U_\sigma \text{diag}(\alpha_1, \dots, \alpha_n) U^{-1}.$$

This implies

$$(3) \quad U_\sigma \text{diag}(\alpha_1, \dots, \alpha_n) U^{-1} V = V_\sigma \text{diag}(\alpha_1, \dots, \alpha_p).$$

This is a homogeneous linear system having  $np$  equations and unknowns  $\alpha_i$ , for  $1 \leq i \leq p$ . Denote by  $\mathcal{SP}$  the solution space of (3). A solution  $\alpha$  is acceptable if and only if  $\alpha > 0$  (the latter implies  $\det(A) \neq 0$  by (2)). These conditions are open conditions, so if there is one such solution then there is an open ball of such solutions of dimension  $q := \dim \mathcal{SP}$ , as well. But we know by Theorem 4 that  $q \leq 1$  for non-decomposable cones. Thus either  $q = 1$ , or  $\sigma \notin \text{Proj}(\mathcal{C})$ . If  $q = 1$ , we can find a nonzero solution  $\alpha$  of (3) and test that  $\pm \alpha > 0$ . If there is no such  $\alpha$ , we conclude that  $\sigma \notin \text{Proj}(\mathcal{C})$ . Otherwise, picking the right sign of  $\alpha$ , we find  $A$  using (2) and conclude that  $\sigma \in \text{Proj}(\mathcal{C})$ .  $\square$

Theorem 6 gives a constructive way to compute  $\text{Proj}(\mathcal{C})$ . Combining with the intermediate subgroup algorithm to compute  $\text{Comb}(\mathcal{C})$  given in Section 5 gives a more practical method to compute  $\text{Proj}(\mathcal{C})$ . An easy situation is when  $\text{Lin}_v(\mathcal{C}) = \text{Comb}(\mathcal{C})$ , which of course implies  $\text{Lin}_v(\mathcal{C}) = \text{Proj}(\mathcal{C})$ .

Let us take  $\alpha_i > 0$ . The group  $\text{Lin}_{\alpha v}(\mathcal{C})$  depends on  $\alpha$  and is a subgroup of  $\text{Proj}(\mathcal{C})$ . By Theorem 4 there exist  $\alpha$  such that  $\text{Lin}_{\alpha v}(\mathcal{C}) = \text{Proj}(\mathcal{C})$  and it is interesting to know when equality occurs. For any  $\alpha > 0$  the group  $\text{Lin}_{\alpha v}(\mathcal{C})$  is a symmetric group acting on  $p$  points which defines an orbit partition  $\text{OP}(\alpha)$  of  $\{1, \dots, p\}$ .

**Theorem 7.** (i) If  $(\alpha_i)_{1 \leq i \leq p}$  and  $(\alpha'_i)_{1 \leq i \leq p}$  are two sets of positive multipliers and  $\text{OP}(\alpha) = \text{OP}(\alpha')$  then  $\text{Lin}_{\alpha v}(\mathcal{C}) = \text{Lin}_{\alpha' v}(\mathcal{C})$

(ii) If  $(\alpha_i)_{1 \leq i \leq p}$  is a set of positive multiplier and  $\text{Lin}_{\alpha v}(\mathcal{C})$  is transitive on  $\{1, \dots, p\}$  then  $\text{Lin}_{\alpha v}(\mathcal{C}) = \text{Proj}(\mathcal{C})$ .

*Proof.* Let us prove (i). We decompose  $\mathcal{C}$  into non-decomposable components  $\mathcal{C}_k$  for  $1 \leq k \leq h$  and denote by  $S_k$  the corresponding subset of  $\{1, \dots, p\}$ . Let us take an orbit  $O$  under  $G_\alpha = \text{Lin}_{\alpha v}(\mathcal{C})$ . If  $x \in O \cap S_k$  and  $H = \text{Stab}_{G_\alpha}(S_k)$  then Theorem 3 giving the expression of the projective symmetry group in terms of a wreath product is also valid for the linear automorphism group. This implies that the orbit of  $x$  under  $H$  is exactly  $O \cap S_k$ . Our assumption  $\text{OP}(\alpha) = \text{OP}(\alpha')$  implies

that  $O \cap S_k$  is also an orbit under  $H' = \text{Stab}_{G_{\alpha'}}(S_k)$ . Furthermore, by the non-decomposability of  $\mathcal{C}_k$  we know that  $\alpha_l$  is determined up to some constant factor for  $l \in O \cap S_k$ . Thus there exists a  $\beta > 0$  such that  $\alpha_l = \beta \alpha'_l$  for  $l \in O \cap S_k$ . This implies that the linear automorphism groups of  $\mathcal{C}_k$  for  $(\alpha_i v_i)_{i \in S_k}$  and  $(\alpha'_i v_i)_{i \in S_k}$  are equal. Since  $\text{OP}(\alpha) = \text{OP}(\alpha')$  we know that the isomorphism type of the component  $\mathcal{C}_k$  under  $G_\alpha$  and  $G_{\alpha'}$  are the same. Since both groups are actually direct products of wreath products they are necessarily equal.

To prove (ii), note that by Theorem 4 there exist some multiplier  $\alpha'$  such that  $G_{\alpha'} = \text{Proj}(\mathcal{C})$ .  $G_\alpha$  is a subgroup of  $G_{\alpha'}$  so  $\text{OP}(\alpha)$  is a partition induced from  $\text{OP}(\alpha')$  by splitting some orbit. But  $\text{OP}(\alpha)$  is reduced to only one component so  $\text{OP}(\alpha) = \text{OP}(\alpha')$  and one concludes.  $\square$

By using item (ii) above one can conclude in some cases that the linear group is actually the projective group.

## 5. COMPUTING $\text{Comb}(\mathcal{C})$

Recall that  $\text{Comb}(\mathcal{C})$  is the maximal symmetry group of a polyhedral cone that preserves the face lattice. For many polyhedral computations, this is the largest group of symmetries that can be exploited. Although no efficient methods are known for the general case, in this section we describe some techniques that can be useful in certain practical computations. The general idea is to construct a “sandwich”  $G_1 \leq \text{Comb}(\mathcal{C}) \leq G_2$  between groups  $G_1$  and  $G_2$  that are easier to compute. We present a technique based on double coset decomposition that can be used to speed up the testing of such a group inclusion for strictness.

We define the group of combinatorial symmetries  $\text{Skel}_k(\mathcal{C})$  to be the group of symmetries preserving the faces of dimension at most  $k$ . In particular  $\text{Skel}_1(\mathcal{C}) = \text{Sym}(p)$ ,  $\text{Skel}_{k+1}(\mathcal{C}) \leq \text{Skel}_k(\mathcal{C})$  and  $\text{Skel}_{n-1}(\mathcal{C}) = \text{Comb}(\mathcal{C})$ .

For a chosen set of generators  $(v_i)_{1 \leq i \leq p}$  and an integer  $k \geq 0$  we have the inclusion

$$(4) \quad \text{Lin}_v(\mathcal{C}) \leq \text{Proj}(\mathcal{C}) \leq \text{Comb}(\mathcal{C}) \leq \text{Skel}_k(\mathcal{C}).$$

Assuming that we know the set  $\mathcal{F}_k$  of  $k$ -dimensional faces, the group  $\text{Skel}_k(\mathcal{C})$  is isomorphic to the automorphism group of a vertex colored graph on  $p + |\mathcal{F}_k|$  vertices. The reason is that if an automorphism preserves all the  $k$ -dimensional faces, then it preserve all the intersections and so all the faces of dimension at most  $k$ . The  $k$ -dimensional faces  $F$  are thus described by the set  $S$  of vertices contained in them and so we can build a bipartite graph on  $p + |\mathcal{F}_k|$  vertices that encodes this relation.

By the chain of inclusions in (4), the group  $\text{Comb}(\mathcal{C})$  is located between two groups which are both automorphism groups of colored graphs. If we can prove that for some  $k_0$  we have  $\text{Lin}_v(\mathcal{C}) = \text{Skel}_{k_0}(\mathcal{C})$  then we conclude that  $\text{Comb}(\mathcal{C}) = \text{Lin}_v(\mathcal{C})$  and we are finished. This is the most common method (see [8, 7]) for computing combinatorial symmetry groups: determine the set of faces of dimension at most  $k_0$ , determine  $\text{Skel}_{k_0}(\mathcal{C})$ , test if the elements of  $\text{Skel}_{k_0}(\mathcal{C})$  are actually in  $\text{Lin}_v(\mathcal{C})$  and if yes obtain  $\text{Skel}_{k_0}(\mathcal{C}) = \text{Comb}(\mathcal{C})$ . But this does not always work since in some cases  $\text{Lin}_v(\mathcal{C}) \neq \text{Comb}(\mathcal{C})$ . One case where it is guaranteed to work

is for simple polytopes, i.e. ones for which every vertex is adjacent to exactly  $n$  other vertices, for these polytopes  $\text{Comb}(\mathcal{C}) = \text{Skel}_2(\mathcal{C})$  [2, 15, 11].

Typically, the number of  $k$ -dimensional faces becomes impractically large for some intermediate values of  $k$ . An alternative method for computing  $\text{Comb}(\mathcal{C})$  is simply to compute the whole set of facets and then compute  $\text{Skel}_{n-1}(\mathcal{C}) = \text{Comb}(\mathcal{C})$  directly. The problem of this method is that the set of facets may be too large for this approach to work and sometimes the facets precisely what we want to compute in the end.

Suppose now that we have a groups  $G_1, G_2$  with  $G_1 \leq \text{Comb}(\mathcal{C}) \leq G_2$ . We now consider a method based on double-coset decomposition to test if these inclusions are strict, possibly replacing  $G_1$  with a larger subgroup of  $\text{Comb}(\mathcal{C})$  in the process.

Let us assume that we have computed the orbits of facets of  $\mathcal{C}$  up to  $G_1$ . The possible methods for doing such a computation are reviewed in [5]. Denote by  $O_1, \dots, O_r$  the orbits of facets, for which we select some representative  $F_1, \dots, F_r$ . They are encoded by their vertex incidence as subsets  $S_1, \dots, S_r \subset \{1, \dots, p\}$ .

The first step is to be able to say whether a permutation  $\sigma$  of the vertex set of  $\mathcal{C}$  does belong to  $\text{Comb}(\mathcal{C})$ . A permutation  $\sigma \in \text{Sym}(p)$  belongs to  $\text{Comb}(\mathcal{C})$  if and only if any image  $\sigma(S_i)$  is in a  $G_1$ -orbit of one of our representatives  $S_j$ . Such in-orbit tests are done using permutation backtrack algorithms [20, 21] that are implemented, for example in GAP [35] and PermLib [39].

Now suppose we are given three groups  $G_1 \subset H \subset G_2$  with  $H$  described by an oracle (e.g. the procedure based on  $G_1$ -orbits just described). We propose an *intermediate subgroup algorithm* for determining an explicit representation for  $H$ . We can do a double coset decomposition of  $G_2$  using the subgroup  $G_1$ :

$$G_2 = \bigcup_{i=1}^s G_1 g_i G_1$$

with  $g_i \in G_2$  and  $G_1 g_i G_1 \cap G_1 g_{i'} G_1 \neq \emptyset$  if and only if  $i = i'$ . Suppose that  $g \in H$ , then since  $G_1 \subset G_2$  for every  $f, f' \in G_1$ , we have  $fgf' \in H$ . So, for a given  $g \in G_2$  either  $G_1 g G_1 \subset H$  or  $G_1 g G_1 \cap H = \emptyset$ . This allows a reduction in the number of oracle calls. Additionally if we found a  $g \in G_2 - G_1$  that belongs to  $H$  then we can replace  $G_1$  by the group generated by  $g$  and  $G_1$  and recompute the double coset decomposition. In particular this method allows us to compute  $\text{Comb}(\mathcal{C})$  without having to iterate over all elements of  $\text{Skel}_{k_0}(\mathcal{C})$  and test whether they belong to  $\text{Comb}(\mathcal{C})$ .

It is not clear how to do much better since in general one needs the facets in order to get  $\text{Comb}(\mathcal{C})$ . The underlying assumption to get good performance using the intermediate subgroup algorithm is that the index  $[G_2 : G_1]$  is not “too large”, i.e.  $[\text{Skel}_{k_0}(\mathcal{C}) : \text{Lin}_v(\mathcal{C})]$  is not “too large”.

## REFERENCES

- [1] R. Baeza, M.I. Icaza, *On the unimodularity of minimal vectors of Humbert forms*, Arch. Math. (Basel) 83 (2004), no. 6, 528–535.

- [2] R. Blind and P. Mani-Levitska, *Puzzles and polytope isomorphism*, Aequationes Math. **34** (1987) 287–297.
- [3] R. Bödi, K. Herr, and M. Joswig *Algorithms for highly symmetric linear and integer programs*, Math. Program., to appear, DOI: 10.1007/s10107-011-0487-6.
- [4] J. Bokowski, G. Ewald and P. Kleinschmidt, *On combinatorial and affine automorphisms of polytopes*, Israel J. Math. **47-2&3** (1984) 123–130.
- [5] D. Bremner, M. Dutour Sikirić and A. Schürmann, *Polyhedral representation conversion up to symmetries*, CRM proceedings & Lecture Notes **48** (2009) 45–72.
- [6] D.A. Cox, J.B. Little and H.K. Schenck, *Toric Varieties*, AMS, 2011.
- [7] A. Deza and M. Deza, *Skeletons of some relatives of the  $n$ -cube*, Optimization theory and its applications in mathematical systems, Surikaiseikenkyusho Kokyuroku No. 899 (1995) 114–120.
- [8] A. Deza, B. Goldengorin, D.V. Pasechnik, *The isometries of the cut, metric and hypermetric cones*, J. Algebraic Combin. **23-3** (2006) 197–203.
- [9] M. Dutour Sikirić, F. Vallentin and A. Schürmann, *Classification of eight-dimensional perfect forms*, Electronic Research Inducement’s of the AMS **13** (2007) 21–32.
- [10] M. Dutour Sikirić, A. Schürmann and F. Vallentin, *Complexity and algorithms for computing Voronoi cells of lattices*, Mathematics of computation **78** (2009) 1713–1731.
- [11] E. Friedman, *Finding a simple polytope from its graph in polynomial time*, Discrete Comput. Geom. **41** (2009) 249–256.
- [12] G. Gévais, *A class of cellulated spheres with non-polytopal symmetries*, Canad. Math. Bull. **52-3** (2009) 366–379.
- [13] I. M. Isaacs, *Character theory of finite groups*, Dover, 1994.
- [14] V. Kaibel and A. Schwartz, *On the complexity of polytope isomorphism problems*, Graphs Comb. **19-2** (2003), 215–230.
- [15] G. Kalai, *A simple way to tell a simple polytope from its graph*, J. Combin. Theory Ser. A **49-2** (1988) 381–383.
- [16] J. Köbler, U. Schöning and J. Torán, *Graph Isomorphism Problem: The Structural Complexity*, Birkhäuser, 1993.
- [17] M. Kreuzer, H. Skarke, *On the classification of reflexive polyhedra*, Commun.Math.Phys. **185** (1997), 495–508.
- [18] M. Kreuzer, H. Skarke, *Classification of reflexive polyhedra in three dimensions*, Adv. Theor. Math. Phys. **2-4** (1998), 853–871.
- [19] M. Kreuzer, H. Skarke, *PALP: a package for analysing lattice polytopes with applications to toric geometry*, Comput. Phys. Comm. 157 (2004), no. 1, 87–106.
- [20] J.S. Leon, *Permutation group algorithms based on partitions. I. Theory and algorithms. Computational group theory, Part 2*, J. Symbolic Comput. 12 (1991), no. 4-5, 533583.
- [21] J.S. Leon, *Partitions, refinements, and permutation group computation. Groups and computation, II* (New Brunswick, NJ, 1995), 123158, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., 28, Amer. Math. Soc., Providence, RI, 1997.
- [22] C.-K. Li and T. Milligan, *Linear preservers of finite reflection groups*, Linear and Multilinear algebra **41** (2003) 49–81.
- [23] L. Liberti, *Reformulations in mathematical programming: automatic symmetry detection and exploitation*, Math. Program. **131** (2012) 273–304.
- [24] F. Margot, *Symmetry in integer linear programming*, in 50 years of integer programming, Springer, 2010, pp. 647–686.
- [25] M. Oswald, *Weighted Consecutive Ones Problems*, dissertation, Universität Heidelberg, 2001.
- [26] A. Paffenholz, G.M. Ziegler, *The  $E_t$ -construction for lattices, spheres and polytopes*, Discrete Comput. Geom. **32-4** (2004) 601–621.
- [27] M.E. Pfetsch and T. Rehn, *Symmetry handling in integer programs revisited*, in preparation.
- [28] W. Plesken and B. Souvignier, *Computing isometries of lattices*, J. Symbolic Comput. **24** (1997) 327–334.



- [29] J.-F. Puget, *Automatic detection of variable and value symmetries*. Lecture Notes in Computer Science vol. 3709, Springer, 2005.
- [30] J.J. Rotman, *An introduction to the theory of groups*, GTM vol. 148, 4th edition, Springer, 1995.
- [31] D. Salvagnin, *A dominance procedure for integer programming*, Master's thesis, University of Padova, 2005.
- [32] A. Schrijver, *Combinatorial optimization. Polyhedra and efficiency. Vol. A, B, C*, Springer, 2003.

#### SOFTWARE

- [33] P. T. Darga, H. Katebi, M. Liffiton, I. Markov and K. Sakallah, *saucy*, <http://vlsicad.eecs.umich.edu/BK/SAUCY/>
- [34] M. Dutour Sikirić, *polyhedral*, <http://drobilica.irb.hr/~mathieu/Polyhedral/>
- [35] The GAP Group, *GAP — Groups, Algorithms, and Permutations*, Version 4.4.6, 2005, <http://www.gap-system.org>
- [36] T. Junttila and P. Kaski, *bliss*, <http://www.tcs.hut.fi/Software/bliss/>
- [37] B.D. McKay, *The nauty program*, <http://cs.anu.edu.au/people/bdm/nauty/>.
- [38] M. Kreuzer, H. Skarke, and others, *PALP: a package for analysing lattice polytopes*, <http://hep.itp.tuwien.ac.at/~kreuzer/CY/CYpalp.html>
- [39] T. Rehn, *Permlib*, <http://www.geometrie.uni-rostock.de/software/>
- [40] T. Rehn and A. Schürmann, *SymPol*, <http://www.geometrie.uni-rostock.de/software/>
- [41] *CPLEX, Mathematical programming technology*, <http://www.ilog.com/products/cplex/>
- [42] *Gurobi, High-end libraries for math programming*, <http://www.gurobi.com/>

#### WEB PAGE

- [43] A. Paffenholz, <http://www.mathematik.tu-darmstadt.de/~paffenholz/data.html>

FACULTY OF COMPUTER SCIENCE, UNIVERSITY OF NEW BRUNSWICK, BOX 4400, FREDERICTON NB, E3B 5A3 CANADA

*E-mail address:* bremner@unb.ca

M. DUTOUR SIKIRIĆ, RUDJER BOSKOVIĆ INSTITUTE, BIJENICKA 54, 10000 ZAGREB, CROATIA

*E-mail address:* mdsikir@irb.hr

SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES, NANYANG TECHNOLOGICAL UNIVERSITY, 21 NANYANG LINK, 637371 SINGAPORE

*E-mail address:* dima@ntu.edu.sg

INSTITUTE OF MATHEMATICS, UNIVERSITY OF ROSTOCK, 18051 ROSTOCK, GERMANY

*E-mail address:* thomas.rehn@uni-rostock.de

INSTITUTE OF MATHEMATICS, UNIVERSITY OF ROSTOCK, 18051 ROSTOCK, GERMANY

*E-mail address:* achill.schuermann@uni-rostock.de